


```
PPPPPPPP  LL      IIIIII  FFFFFFFFFF  000000  RRRRRRRR  MM      MM  AAAAAA  TTTTTTTTTT
PPPPPPPP  LL      IIIIII  FFFFFFFFFF  000000  RRRRRRRR  MM      MM  AAAAAA  TTTTTTTTTT
PP      PP  LL      II      FF      00      00  RR      RR  MMMM  MMMM  AA      AA  TT
PP      PP  LL      II      FF      00      00  RR      RR  MMMM  MMMM  AA      AA  TT
PP      PP  LL      II      FF      00      00  RR      RR  MM  MM  MM  AA      AA  TT
PPPPPPPP  LL      II      FFFFFFFF  00      00  RRRRRRRR  MM      MM  AA      AA  TT
PPPPPPPP  LL      II      FFFFFFFF  00      00  RRRRRRRR  MM      MM  AA      AA  TT
PP      LL      II      FF      00      00  RR  RR  MM      MM  AAAAAAAAAA  TT
PP      LL      II      FF      00      00  RR  RR  MM      MM  AAAAAAAAAA  TT
PP      LL      II      FF      00      00  RR  RR  MM      MM  AA      AA  TT
PP      LL      II      FF      00      00  RR  RR  MM      MM  AA      AA  TT
PP      LL      II      FF      00      00  RR  RR  MM      MM  AA      AA  TT
PP      LLLLLLLLLL  IIIIII  FF      000000  RR      RR  MM      MM  AA      AA  TT
PP      LLLLLLLLLL  IIIIII  FF      000000  RR      RR  MM      MM  AA      AA  TT

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```

```
0000 1      .title pliformat
0000 2      .ident /1-006/
0000 3
0000 4
0000 5
0000 6
0000 7
0000 8
0000 9
0000 10     *****
0000 11     *
0000 12     *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 13     *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 14     *  ALL RIGHTS RESERVED.
0000 15     *
0000 16     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 17     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 18     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 19     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 20     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 21     *  TRANSFERRED.
0000 22     *
0000 23     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 24     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 25     *  CORPORATION.
0000 26     *
0000 27     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 28     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 29     *
0000 30     *****
0000 31
0000 32     ++
0000 33     facility:
0000 34
0000 35         VAX/VMS PL1 runtime library
0000 36
0000 37     abstract:
0000 38
0000 39         This module contains the pl1 runtime routines for getting the next
0000 40         format item in a format list.
0000 41
0000 42
0000 43     author: c. spitz 28-nov-79
0000 44
0000 45     modified:
0000 46
0000 47         1-002   Chip Nylander   7-September-1982
0000 48
0000 49         Modified GETCOL to conform to ANSI X3.53 page 259
0000 50         step 1.2.3.2.3: if a column request cannot be satisfied for
0000 51         any reason, do an implicit getskip; if the column request can
0000 52         now be satisfied, perform the column positioning, otherwise
0000 53         do nothing.
0000 54
0000 55
0000 56         1-003   Bill Matthews   29-September-1982
0000 57
```

```
; Edit CGN1006
; Edit CGN1005
; Edit CGN1004
; Edit WHM1003
```



```
0000 58 : Invoke macros $defdat and rtshare instead of $defopr and share.
0000 59 :
0000 60 : 1-004 Chip Nylander 03-February-1983
0000 61 :
0000 62 : Save the parent frame pointer in R1 instead of R0 when calling
0000 63 : an expression routine.
0000 64 :
0000 65 : 1-005 Chip Nylander 23-February-1983
0000 66 :
0000 67 : Make fixed-point edited output of floating binary values
0000 68 : round instead of truncate, per the ANSI Standard and our
0000 69 : own published documentation.
0000 70 :
0000 71 : 1-006 Chip Nylander 08-August-1983
0000 72 :
0000 73 : Solve problem with uplevel references to automatic variables
0000 74 : in remote formats by using the parent pointer of the frame
0000 75 : containing the remote format. Use the parent pointer instead
0000 76 : of -4(fp) for all vfe calls.
0000 77 :
0000 78 :--
0000 79 :
0000 80 :
0000 81 : external definitions
0000 82 :
0000 83 : $defstr ;define stream block offsets
0000 84 : $defdat ;define data types
0000 85 : $defcvtind ;define convert indices
0000 86 : $deffcb ;define file control block
0000 87 : $defpic ;define picture node offsets
0000 88 : $sdef ;define stack offsets
0000 89 :
0000 90 :
0000 91 : local data
0000 92 :
0000 93 :
0000 94 : rtshare ;sharable
0000 95 :
0000 96 bformattab: ;table of chars for B-radix conversion
0000 97 .byte ^a\0\,^a\1\ ;entries for B1
0002 98 .byte ^a\0\,^a\2\,^a\1\,^a\3\ ;entries for B2
0006 99 .byte ^a\0\,^a\4\,^a\2\,^a\6\ ;entries for B3
000A 100 .byte ^a\1\,^a\5\,^a\3\,^a\7\ ;
000E 101 .byte ^a\0\,^a\8\,^a\4\,^a\C\ ;entries for B4
0012 102 .byte ^a\2\,^a\A\,^a\6\,^a\E\ ;
0016 103 .byte ^a\1\,^a\9\,^a\5\,^a\D\ ;
001A 104 .byte ^a\3\,^a\B\,^a\7\,^a\F\ ;
001E 105
001E 106 :++
001E 107 : pli$$getfmt_r6
001E 108 :
001E 109 : functional description:
001E 110 : control formats are processed and the next item is transmitted from the
001E 111 : file buffer via edit directed input. for data formats, the general
001E 112 : flow is: the compiled code jsb's to pli$getc*** routine. that
001E 113 : routine saves the source address and precision and jsb's to this
001E 114 : routine to get the next input item. this routine processes interceding
```

```
33 31 31 30
36 32 34 30
37 33 35 31
43 34 38 30
45 36 41 32
44 35 39 31
46 37 42 33
```

```
001E 115 : control formats until a data format is encountered. the data format
001E 116 : evaluates its parameters and gets the proper number of characters by
001E 117 : jsb'ing to pli$$getnedi. pli$$getnedi returns a character string
001E 118 : in the field area of the current format. the data format routine then
001E 119 : converts this character string to the a temporary, whose data type is
001E 120 : based on the format. it returns with the address, precision and data
001E 121 : type of this temporary. the pli$gete*** routine then restores the
001E 122 : target information and calls pli$cvrt_cg_r3 to finish processing.
001E 123 : note that the common control formats for input and output are located
001E 124 : in this section. all output control formats MUST PRESERVE R5, which is
001E 125 : used to store the offset of unaligned bit sources.
001E 126 :
001E 127 : inputs:
001E 128 :   r11 - address of stream block
001E 129 :   ap - address of fcb
001E 130 : outputs:
001E 131 :   r0 - address of field in stream block
001E 132 :   r1 - precision / scale of temp in stream block
001E 133 :   r4 - case index of temp as the source to any
001E 134 : side effects:
001E 135 :   r0-r6 are destroyed
001E 136 :--
001E 137 :
001E 138 pli$$getfmt r6::
50 04 BB 90 001E 139   movb   @str_l fp(r11),r0      ;get format type
04 AB D6 0022 140   incl   str_l fp(r11)      ;update format pointer
0025 141   case   type=5,r0,limit=#1,< - ;case on format type
0025 142   getbiter, - ;1 byte constant iteration
0025 143   getwiter, - ;2 word constant iteration
0025 144   getliter, - ;3 long constant iteration
0025 145   invfrm, - ;4 pc relative iter (invalid)
0025 146   getexpriter, - ;5 expression iteration (Version 1)
0025 147   geteof, - ;6 end of format
0025 148   getexpriter_v2, - ;7 expression iteration (Version 2)
0025 149   invfrm, - ;8 invalid format
0025 150   invfrm, - ;9 invalid format
0025 151   geta, - ;10 alphanumeric format
0025 152   getb1, - ;11 bit (1) format
0025 153   getb1, - ;12 bit 1 format
0025 154   getb2, - ;13 bit 2 format
0025 155   getb3, - ;14 bit 3 format
0025 156   getb4, - ;15 bit 4 format
0025 157   getcol, - ;16 column format
0025 158   getcol, - ;17 column format
0025 159   gete, - ;18 exp format
0025 160   getf, - ;19 fixed format
0025 161   invfrm, - ;20 line format invalid for get
0025 162   getp, - ;21 picture format
0025 163   invfrm, - ;22 page format invalid for get
0025 164   getr, - ;23 remote format (PL/I version 1)
0025 165   getskip, - ;24 skip format
0025 166   invfrm, - ;25 tab format invalid for get
0025 167   getx, - ;26 blank format
0025 168   invfrm, - ;27 left paren (no longer used)
0025 169   getrparen, - ;28 right paren
0025 170   getr_v2> ;29 remote format (PL/I version 2)
0063 171
```



```
0136 31 0063 172      brw      invfrm      ;none of the above, invalid format
      0066 173
      0066 174      ; process an iteration factor. the iteration factor is stored on the format
      0066 175      ; stack as a count and the address of its first item. if the iteration factor
      0066 176      ; is less than or equal to 0, we will skip the format item(s) between the
      0066 177      ; iteration and its matching right paren.
      0066 178
      0066 179 getbiter:      ;byte constant iteration
      0066 180      pushab    pli$$getfmt_r6      ;set return addr
      0069 181      brb      biter      ;cont in common
      006B 182 putbiter:      ;
      006B 183      pushab    pli$$putfmt_r6      ;set return addr
      0071 184 biter:      cvtbl    @str_l_fp(r11),r1      ;get iteration count
      0075 185      incl     str_l_fp(r11)      ;update format pointer
      0078 186      brb      getitercom      ;cont in common
      007A 187
      007A 188 getwiter:      ;word constant iteration
      007A 189      pushab    pli$$getfmt_r6      ;set return addr
      007D 190      brb      witer      ;cont in common
      007F 191 putwiter:      ;
      007F 192      pushab    pli$$putfmt_r6      ;set return addr
      0085 193 witer:      cvtwl    @str_l_fp(r11),r1      ;get iteration count
      0089 194      addl     #2,str_l_fp(r11)      ;update format pointer
      008D 195      brb      getitercom      ;cont in common
      008F 196
      008F 197 getliter:      ;long constant iteration
      008F 198      pushab    pli$$getfmt_r6      ;set return addr
      0092 199      brb      liter      ;cont in common
      0094 200 putliter:      ;
      0094 201      pushab    pli$$putfmt_r6      ;set return addr
      009A 202 liter:      movl     @str_l_fp(r11),r1      ;get iteration count
      009E 203      addl     #4,str_l_fp(r11)      ;update format pointer
      00A2 204      brb      getitercom      ;cont in common
      00A4 205
      00A4 206 getexpriter_v2:      ;expression iteration (Version 2)
      00A4 207      pushab    pli$$getfmt_r6      ;set return addr
      00A8 208      brb      exiter_v2      ;cont in common
      00AA 209 putexpriter_v2:      ;
      00AA 210      pushab    pli$$putfmt_r6      ;set return addr
      00B0 211 exiter_v2:      ;
      00B0 212      movl     str_l_parent(r11),r1      ;get parent frame pointer
      00B4 213      brb      exiter_common      ;join common code
      00B6 214
      00B6 215 getexpriter:      ;expression iteration
      00B6 216      pushab    pli$$getfmt_r6      ;set return addr
      00BA 217      brb      exiter      ;cont in common
      00BC 218 putexpriter:      ;
      00BC 219      pushab    pli$$putfmt_r6      ;set return addr
      00C2 220 exiter:      movl     -4(fp),r1      ;get parent frame pointer
      00C6 221 exiter_common:      ;
      00C6 222      movl     @str_l_fp(r11),r0      ;get rel addr
      00CA 223      addl     #4,str_l_fp(r11)      ;update format pointer
      00CE 224      addl     str_l_fp(r11),r0      ;get absolute addr
      00D2 225      calls     #0,r0      ;call the routine
      00D5 226      movl     r0,r1      ;set iteration factor
      00D8 227
      00D8 228 getitercom:      ;process iteration factor
```

```
50 52 68 D0 00D8 229 movl str_l_sp(r11),r2 ;get format stack pointer
    0410 CB 9E 00DB 230 movab <str_l_stack_end+8>(r11),r0 ;get last place for an iteration
    52 50 D1 00E0 231 cmpl r0,r2 ;is there room for another iteration?
    OA 1B 00E3 232 blequ r0,r2 ;if lequ, yes continue
50 00000000 8F D0 00E5 233 movl #plis_formatovfl,r0 ;set format stack overflow
    0437 31 00EC 234 brw fail ;and fail
    72 04 AB D0 00EF 235 10$: movl str_l_fp(r11),-(r2) ;push fp on stack
    72 51 D0 00F3 236 movl r1,-(r2) ;push iter count on stack
    6B 52 D0 00F6 237 movl r2,str_l_sp(r11) ;store stack pointer
    4A 62 F4 00F9 238 sobgeq (r2),30$ ;do an iteration
    00FC 239 ; the format iteration is < 0, so we must skip all format items until the
    00FC 240 ; matching right paren is found.
    62 D4 00FC 241 clrl (r2) ;skip this iteration, clear paren count
50 04 BB 90 00FE 242 20$: movb @str_l_fp(r11),r0 ;get next format
    04 AB D6 0102 243 incl str_l_fp(r11) ;increment format pointer
    0105 244 type=5,r0,limit=#1,< - ;case on format type
    0105 245 70$, - ;1 byte iter
    0105 246 80$, - ;2 word iter
    0105 247 90$, - ;3 long iter
    0105 248 invfrm, - ;4 pc rel cons
    0105 249 90$, - ;5 expression iter (Version 1)
    0105 250 invfrm, - ;6 end of format (not expected)
    0105 251 90$, - ;7 expression iter (Version 2)
    0105 252 invfrm, - ;8 unused
    0105 253 invfrm, - ;9 unused
    0105 254 50$, - ;10 a
    0105 255 50$, - ;11 b1
    0105 256 50$, - ;12 b1
    0105 257 50$, - ;13 b2
    0105 258 50$, - ;14 b3
    0105 259 50$, - ;15 b4
    0105 260 50$, - ;16 col
    0105 261 50$, - ;17 col
    0105 262 40$, - ;18 e
    0105 263 40$, - ;19 f
    0105 264 50$, - ;20 lin
    0105 265 50$, - ;21 pic
    0105 266 20$, - ;22 page
    0105 267 50$, - ;23 rem (PL/I version 1)
    0105 268 50$, - ;24 skip
    0105 269 50$, - ;25 tab
    0105 270 50$, - ;26 x
    0105 271 invfrm, - ;27 left paren
    0105 272 60$, - ;28 right paren
    0105 273 45$> ;29 rem (PL/I version 2)
    0143 274
    0056 31 0143 275 brw invfrm ;invalid format
    05 05 0146 276 30$: rsb ;process next format item
    0356 30 0147 277 40$: bsbw get_format_parm ;get first parm
    0353 30 014A 278 45$: bsbw get_format_parm ;get second parm
    0350 30 014D 279 50$: bsbw get_format_parm ;get last parm
    AC 11 0150 280 20$ ;go again
    62 D7 0152 281 60$: decl (r2) ;decrement paren count
    A8 18 0154 282 bgeq 20$ ;if geq, then go again
    6B 08 C0 0156 283 addl #8,str_l_sp(r11) ;clean stack
    05 05 0159 284 rsb ;process next format item
    04 AB D6 015A 285 70$: incl str_l_fp(r11) ;skip iteration
```



```
04 AB 0A 11 015D 286 brb 100$ ;continue
04 AB 02 C0 015F 287 80$: addl #2,str_l_fp(r11) ;skip iteration
04 AB 04 11 0163 288 brb 100$ ;continue
04 AB 04 C0 0165 289 90$: addl #4,str_l_fp(r11) ;skip iteration
62 D6 0169 290 100$: incl (r2) ;increment paren count
FF90 31 016B 291 brw 20$ ;go again
016E 292
016E 293 ; end of format - if processing remote format, return to 'caller'. otherwise
016E 294 ; repeat format.
FEAC CF 9F 016E 295 geteof: pushab pli$$getfmt_r6 ;set return addr
06 11 0172 296 brb comeof ;cont in com
0000053C'EF 9F 0174 297 puteof: pushab pli$$putfmt_r6 ;set return addr
50 0C04 CB 9E 017A 298 comeof: movab str_l_stack(r11),r0 ;get addr of top of stack
50 6B D1 017F 299 cmpl str_l_sp(r11),r0 ;anything on the stack?
07 1F 0182 300 blssu 10$ ;if lssu, yes, its end of remote
04 AB 0C04 CB D0 0184 301 movl str_l_stack(r11),str_l_fp(r11) ;restart the format
05 018A 302 rsb ;go again
08 AB 00 BB D0 018B 303 10$: movl @str_l_sp(r11),str_l_parent(r11) ;reset parent pointer
6B 04 C0 0190 304 addl #4,str_l_sp(r11) ;
04 AB 00 BB D0 0193 305 movl @str_l_sp(r11),str_l_fp(r11) ;reset format pointer
6B 04 C0 0198 306 addl #4,str_l_sp(r11) ;clean stack
05 019B 307 rsb ;go again
019C 308
50 00000000'8F D0 019C 309 invfrm: movl #pli$_invformat,r0 ;set invalid format
0380 31 01A3 310 brw fail ;and fail
01A6 311
50 00000000'8F D0 01A6 312 invfrmprm: movl #pli$_invfmtprm,r0 ;set invalid format parameter
0376 31 01AD 314 brw fail ;and fail
01B0 315
50 00000000'8F D0 01B0 316 invstrfmt: movl #pli$_invstrfmt,r0 ;set invalid stream format
036C 31 01B7 317 brw fail ;and fail
01BA 318 ; a format, input. get the width, get that number of chars and return.
02E3 30 01BA 319 geta: bsbw get_format_parm ;get the parameter
E7 15 01BD 320 bleq invfrmprm ;if leg, then invalid format
50 51 D0 01BF 321 movl r1,r0 ;set width
00000000'GF 16 01C2 322 jsb g^pli$$getnedi_r6 ;get the item
54 2D D0 01C8 323 movl #cvt_k_src_char,r4 ;set case index
05 01CB 324 rsb ;return
01CC 325
01CC 326 ; b format, input
01CC 327 ; set the radix factor
01  DD 01CC 328 getb1: pushl #1 ;push radix
0A 11 01CE 329 brb getb ;continue in common
02  DD 01D0 330 getb2: pushl #2 ;push radix
06 11 01D2 331 brb getb ;continue in common
03  DD 01D4 332 getb3: pushl #3 ;push radix
02 11 01D6 333 brb getb ;continue in common
04  DD 01D8 334 getb4: pushl #4 ;push radix
01DA 335 ; get the width and that number of characters
02C3 30 01DA 336 getb: bsbw get_format_parm ;get the parameter
C7 15 01DD 337 bleq invfrmprm ;if leg, then invalid format
50 51 D0 01DF 338 movl r1,r0 ;set width
00000000'GF 16 01E2 339 jsb g^pli$$getnedi_r6 ;get the item
18 AB 51 20 3B 01E8 340 ; skip leading blanks. there must be at least 1 non-blank.
0A 12 01ED 341 skpc #^x20,r1,str_b_field(r11) ;skip leading blanks
342 bneq 20$ ;if neq, non-blank found
```



```
50 00000000'8F D0 01EF 343 10$: movl #pli$_cnverr,r0 ;set conversion error
      032D 31 01F6 344      brw fail ;and fail
      56 50 D0 01F9 345 20$: movl r0,r6 ;save length left
      55 51 D0 01FC 346      movl r1,r5 ;save addr of 1st non-blank
61 50 20 3A 01FF 347 ; locate trailing blanks. we won't convert them
      09 13 0203 348      locc #^x20,r0,(r1) ;find next blank
      56 50 C2 0205 349      beql 30$ ;if eql, not found, cont
61 50 20 3B 0208 350      subl r0,r6 ;get new length for string
      E1 12 020C 351      skpc #^x20,r0,(r1) ;anything left other than blanks?
      020E 352      bneq 10$ ;if neq, then yes, error
18 AB 65 56 28 020E 353 ; copy the non-blank chars to the beginning of the field
      10 AB 53 D0 0213 354 30$: movc3 r6,(r5),str_b_field(r11) ;copy to beginning of field
      50 8ED0 0217 355      movl r3,str_l_flg_pt(r11) ;set field pointer
54 00000000'GF 16 021A 356 ; convert the chars to a bit string based on the radix factor
      00000048 8F D0 0220 357      popl r0 ;restore radix
      05 0227 358      jsb g^plis$chrbitn_r6 ;convert to a bit string
      0228 359      movl #cvt_k_src_abif,r4 ;set case index
      0228 360      rsb ;return
      0228 361
      0228 362 ; column format, input
      0228 363 ; if the requested column is after current column, and before the end of the
      0228 364 ; line, we just position the buffer pointer to the requested place.
      0228 365 ; otherwise, perform a getskip; if the column can now be positioned as
      0228 366 ; requested, then do so, otherwise give up.
      0228 367
03 OC AC 17 E1 0228 368 getcol: bbc #atr_v_string,fcbl_attr(ap),5$ ;if string i/o
      FF80 31 022D 369      brw invstrfmt ;fail with invalid string format
      026D 30 0230 370 5$: bsbw get_format_parm ;get the parameter
      07 14 0233 371      bgtr 20$ ;if gtr, cont
      03 13 0235 372      beql 10$ ;if eql, use 1
      FF6C 31 0237 373      brw invfrmpm ;its lss, invalid format
      51 D6 023A 374 10$: incl r1 ;use 1 instead of 0
      52 D4 023C 375 20$: clrl r2 ;say that this is first time through
      2E AC 51 B1 023E 376      cmpw r1,fcbl_w_column(ap) ;already past requested column?
      1D 19 0242 377      blss 30$ ;if lss, then yes
50 51 2E AC A3 0244 378 25$: subw3 fcbl_w_column(ap),r1,r0 ;get number to move
      2E AC 51 B0 0249 379      movw r1,fcbl_w_column(ap) ;update column
      50 50 3C 024D 380      movzwl r0,r0 ;make it long
53 1C AC 50 C1 0250 381      addl3 r0,fcbl_buf_pt(ap),r3 ;make updated buffer pointer
      18 AC 53 D1 0255 382      cmpl r3,fcbl_buf_end(ap) ;past end of this line?
      06 18 0259 383      bgeq 30$ ;if geq, then yes
      1C AC 53 D0 025B 384      movl r3,fcbl_buf_pt(ap) ;update buffer pointer
      1D 11 025F 385      brb 40$ ;exit
      52 D5 0261 386 30$: tstl r2 ;is this first time through?
      19 14 0263 387      bgtr 40$ ;if no, give up trying
OC AC 02000000 8F CA 0265 388      bicl #atr_m_virgin,fcbl_attr(ap) ;deflower file (so we don't skip
      53 51 D0 026D 389      ; first record)
      00000000'GF 16 0270 390      movl r1,r3 ;save requested column
      51 53 D0 0276 391      jsb g^plis$getskp1_r2 ;do a skip
      52 01 D0 0279 392      movl r3,r1 ;restore request
      C6 11 027C 393      movl #1,r2 ;say that this is second time through
      FD9D 31 027E 394      brb 25$ ;go try to position on new line
      0281 395 40$: brw plis$getfmt_r6 ;go again
      0281 396
      0281 397 ; e format, input
      0281 398 ; get the parameters. w,d and s are supplied, but s is ignored.
      021C 30 0281 399 gete: bsbw get_format_parm ;get width
```

```
03 18 0284 400 bgeq 10$ ;if geq, cont
FF1D 31 0286 401 brw invfrmprm ;if lss, then invalid format
03 12 0289 402 10$: bneq 20$ ;if neq, then cont
003C 31 028B 403 brw zero ;make result zero
52 51 D0 028E 404 20$: movl r1,r2 ;save width
020C 30 0291 405 bsbw get_format_parm ;get fractional digits
03 18 0294 406 bgeq 30$ ;if geq, cont
FF0D 31 0296 407 brw invfrmprm ;if lss, then invalid format
54 51 D0 0299 408 30$: movl r1,r4 ;set frac digits for pli$fchrfltd_r6
0201 30 029C 409 bsbw get_format_parm ;get scale factor
50 52 D0 029F 410 movl r2,r0 ;set width
02A2 411 ; get the required number of chars
00000000'GF 16 02A2 412 jsb g^pli$$getnedi_r6 ;get the field
0838 30 02A8 413 bsbw charfltctx ;get the float context
52 18 AB 9E 02AB 414 movab str_b_field(r11),r2 ;addr field as target
7E 52 7D 02AF 415 movq r2,=(sp) ;save destination
00000000'GF 00 FB 02B2 416 calls #0,g^pli$fchrfltd_r6 ;convert fractioned char to float dec
50 8E 7D 02B9 417 movq (sp)+,r0 ;use destination of cvt as src
54 24 7D 02BC 418 movl #cvt_k_src_fltd,r4 ;set case index for fltd src
05 02BF 419 rsb ;return
02C0 420
02C0 421 .enabl lsb
02C0 422 ; f format, input
02C0 423 ; get w,d,s. s is ignored
01DD 30 02C0 424 getf: bsbw get_format_parm ;get width
03 18 02C3 425 bgeq 10$ ;if geq, cont
FEDE 31 02C5 426 brw invfrmprm ;its lss, invalid format
13 12 02C8 427 10$: bneq 20$ ;if neq, then cont
01D3 30 02CA 428 zero: bsbw get_format_parm ;get next parm
01D0 30 02CD 429 bsbw get_format_parm ;get last parm
50 18 AB 9E 02D0 430 movab str_b_field(r11),r0 ;set addr of result
60 60 D4 02D4 431 clrl (r0) ;clear result
51 1F D0 02D6 432 movl #31,r1 ;set precision
54 09 D0 02D9 433 movl #cvt_k_src_fixb,r4 ;set case index for fixb
05 02DC 434 rsb ;return
52 51 D0 02DD 435 20$: movl r1,r2 ;save width
01BD 30 02E0 436 bsbw get_format_parm ;get fractional digits
03 18 02E3 437 bgeq 30$ ;if geq, ok
FEDE 31 02E5 438 brw invfrmprm ;its lss, invalid format
53 51 D0 02E8 439 30$: movl r1,r3 ;save fractional digits
01B2 30 02EB 440 bsbw get_format_parm ;get scale factor
50 52 D0 02EE 441 movl r2,r0 ;set width
02F1 442 ; get the required number of chars
00000000'GF 16 02F1 443 jsb g^pli$$getnedi_r6 ;get the field
56 50 D0 02F7 444 movl r0,r6 ;save start addr
52 51 D0 02FA 445 movl r1,r2 ;save length read
02FD 446 ; if there is no decimal point in the input, we use the specified d to imply one
60 51 2E 3A 02FD 447 locc #^x2e,r1,(r0) ;find the decimal point
59 13 0301 448 beql 70$ ;if eql, then use fractional digits
55 51 D0 0303 449 movl r1,r5 ;save addr of point
0306 450 ; make sure there is nothing but trailing blanks
61 50 20 3A 0306 451 locc #^x20,r0,(r1) ;find trailing blank
13 13 030A 452 beql 40$ ;if eql, then none
52 50 C2 030C 453 subl r0,r2 ;get new length of field
61 50 20 3B 030F 454 skpc #^x20,r0,(r1) ;anything left other than blanks?
0A 13 0313 455 beql 40$ ;if eql, then no, ok
50 00000000'8F D0 0315 456 movl #pli$_cnverr,r0 ;set conversion error
```



```
0207 31 031C 457 brw fail ;and fail
031F 458 ; pli$charfixd_r6 allows an exponent, but f format does not. we will append
031F 459 ; an exponent of 0, which will cause charfixd to signal error if the input
031F 460 ; already has an exponent.
50 52 56 C1 031F 461 40$: addl3 r6,r2,r0 ;get addr of end of field
53 50 53 C3 0323 462 subl3 r5,r0,r3 ;get number of fractional digits
53 53 53 D7 0327 463 decl r3 ;
60 3045 8F B0 0329 464 movw #^x3045,(r0) ;append 'E0', (its not allowed in f)
51 52 02 C1 032E 465 addl3 #2,r2,r1 ;set length for convert
1F 52 D1 0332 466 cmpl r2,#31 ;length > max fixd prec?
53 52 03 15 0335 467 bleq 50$ ;if leq, then no, cont
53 53 08 78 0337 468 movl #31,r2 ;use max prec
53 53 52 88 033A 469 50$: ashl #8,r3,r3 ;set scale of temp = # digs in frac
53 53 53 DD 0341 470 bisb r2,r3 ;put in the prec
50 18 AB 9E 0343 471 pushl r3 ;save prec,scale
52 50 D0 0347 472 movab str_b_field(r11),r0 ;set addr of src
00000000'GF 00 FB 034A 473 movl r0,r2 ;set addr of dst
51 8ED0 0351 474 calls #0,g^pli$charfixd_r6 ;convert to fixd
50 18 AB 9E 0354 475 popl r1 ;restore prec,scale
54 1B D0 0358 476 60$: movab str_b_field(r11),r0 ;set addr of temp
54 1B D0 035B 477 movl #cvf_E_src_fixd,r4 ;set case index
54 1B D0 035B 478 rsb ;return
54 1B D0 035C 479 ; there was no decimal point in the input string, so we will convert to a non-
54 1B D0 035C 480 ; scaled fixd, and fix up the scale after the conversion.
50 18 AB 9E 035C 481 70$: movab str_b_field(r11),r0 ;get addr of field
60 52 20 3B 0360 482 skpc #^x20,r2,(r0) ;skip leading blanks
61 50 20 3A 0364 483 locc #^x20,r0,(r1) ;find trailing blank
61 50 18 13 0368 484 beql 90$ ;if eql, no trail blanks, cont
61 50 51 DD 036A 485 pushl r1 ;save start of blanks
61 50 52 C2 036C 486 subl r0,r2 ;don't count blanks in len
61 50 20 3B 036F 487 skpc #^x20,r0,(r1) ;skip trail blanks.
50 00000000'8F D0 0373 488 beql 80$ ;if eql, ok
50 00000000'8F D0 0375 489 movl #pli$_cnverr,r0 ;set conversion error (non blank found)
50 01A7 31 037C 490 brw fail ;and fail
61 3045 8F B0 037F 491 80$: popl r1 ;get start of blanks
51 52 02 C1 0382 492 90$: movw #^x3045,(r1) ;append 'E0' (its not allowed in f)
51 52 02 C1 0387 493 addl3 #2,r2,r1 ;set len of src
51 52 02 D1 038B 494 cmpl r2,#31 ;length > max fixd prec?
51 52 03 15 038E 495 bleq 100$ ;if leq, then no, cont
54 52 1F D0 0390 496 movl #31,r2 ;use max prec
54 53 08 78 0393 497 100$: ashl #8,r3,r4 ;set scale = numb digs in frac
54 54 52 88 0397 498 bisb r2,r4 ;set len
54 54 54 DD 039A 499 pushl r4 ;save prec,scale
50 18 AB 9E 039C 500 movab str_b_field(r11),r0 ;set addr of src
53 52 D0 03A0 501 movl r2,r3 ;set len of dst
52 50 D0 03A3 502 movl r0,r2 ;set addr of dst
00000000'GF 00 FB 03A6 503 calls #0,g^pli$charfixd_r6 ;convert to fixd
51 8ED0 03AD 504 popl r1 ;restore prec,scale
54 11 A2 03B0 505 brb 60$ ;continue
54 11 A2 03B2 506 .dsabl lsb
54 11 A2 03B2 507
54 11 A2 03B2 508 ; picture format, input
54 11 A2 03B2 509 ; get the addr of the picture descriptor
00EB 30 03B2 510 getp: bsbw get_format_parm ;get the parm
03 12 03B5 511 bneq 10$ ;if neq, cont
FDE2 31 03B7 512 brw invfrm ;
56 51 D0 03BA 513 10$: movl r1,r6 ;save picture descr addr
```

```
50 04 A1 9A 03BD 514 ; get the required chars
00000000'GF 16 03BD 515     movzbl pic$b byte_size(r1),r0 ;set size to read
                                03C1 516     jsb     g^pli$$getnedi_r6 ;get the field
                                03C7 517 ; validate the picture. note that p format requires that the chars read be
                                03C7 518 ; a valid picture string. this differs from list input of a picture variable
                                03C7 519 ; where the input must be a valid fixed decimal number.
                                520     pushl   r0 ;set addr
                                521     pushl   r1 ;set length read
                                522     pushl   r6 ;set picture desc addr
00000000'GF 03 FB 03CD 523     calls    #3,g^pli$valid_pic ;validate picture
0A 50 E8 03D4 524     blbs     r0,20$ ;if lbs, cont
50 00000000'8F D0 03D7 525     movl     #pli$_cnvrr,r0 ;set conversion error
0145 31 03DE 526     brw     fail ;and fail
50 18 AB 9E 03E1 527 20$:     movab    str_b_field(r11),r0 ;set addr
51 56 D0 03E5 528     movl     r6,r1 ;set pic desc addr
54 00 D0 03E8 529     movl     #cvt_k_src_pic,r4 ;set src data type
05 03EB 530     rsb ;return
03EC 531
03EC 532 ; version 2 remote format. a remote format is processed by using the nesting
03EC 533 ; level difference passed as the first format param to calculate the parent
03EC 534 ; pointer of the remote format. this calculated parent pointer is then set
03EC 535 ; info r1 for all vfe calls that occur in the remote format, and the vfes
03EC 536 ; use r1 for uplevel references to automatic variables.
FC2E CF 9F 03EC 537 getr_v2:pushab pli$$getfmt_r6 ;set return addr
06 11 03F0 538     brb     comr_v2 ;cont in common
0000053C'EF 9F 03F2 539 putr_v2:pushab pli$$putfmt_r6 ;set return addr
00A5 30 03F8 540 comr_v2:bsbw get_format_parm ;get nesting level relative to referencer
03 18 03FB 541     bgeq    10$ ;if geg, continue
FD9C 31 03FD 542     brw     invfrm ;else invalid format
53 08 AB D0 0400 543 10$:     movl     str_l_parent(r11),r3 ;get parent pointer of referencer
51 D7 0404 544 20$:     decl     r1 ;decrement relative nesting level
16 19 0406 545     blss     remcom ;if lss then have correct parent pointer
53 0C A3 D0 0408 546     movl     sf$L_save_fp(r3),r3 ;else get next higher parent pointer
F6 11 040C 547     brb     20$ ;and go back
040E 548
040E 549
040E 550 ; remote format. a remote format is processed by pushing the address of the
040E 551 ; next item in the original format onto the format stack. when the remote
040E 552 ; format's end of format is encountered, this address is popped, and control
040E 553 ; returns to the original format.
FC0C CF 9F 040E 554 getr: pushab pli$$getfmt_r6 ;set return addr
06 11 0412 555     brb     comr ;cont in common
0000053C'EF 9F 0414 556 putr: pushab pli$$putfmt_r6 ;set return addr
53 08 AB D0 041A 557 comr: movl str_l_parent(r11),r3 ;pickup default parent
007F 30 041E 558 remcom: bsbw get_format_parm ;get the remote format
03 12 0421 559     bneq    10$ ;if neg, continue
FD76 31 0423 560     brw     invfrm ;invalid format
50 0410 CB 9E 0426 561 10$:     movab    <str_l_stack_end+8>(r11),r0 ;get addr of last place for remote
52 6B D0 042B 562     movl     str_l_sp(r11),r2 ;get format stack pointer
52 50 D1 042E 563     cmpl     r0,r2 ;room for this remote?
0A 18 0431 564     blequ    20$ ;if lequ, then yes
50 00000000'8F D0 0433 565     movl     #pli$_formatovfl,r0 ;set format stack overflow
00E9 31 043A 566     brw     fail ;and fail
72 04 AB D0 043D 567 20$:     movl     str_l_fp(r11),-(r2) ;push addr of next item in this format
72 08 AB D0 0441 568     movl     str_l_parent(r11),-(r2) ;push parent pointer for this format
6B 52 D0 0445 569     movl     r2,str_l_sp(r11) ;store stack pointer
04 AB 51 D0 0448 570     movl     r1,str_l_fp(r11) ;set format pointer to remote format
```



```
08 AB 53 D0 044C 571      movl    r3,str_l_parent(r11)      ;set parent pointer to remote format
05      0450 572      rsb                    ;go with remote format
0451 573
0451 574      ; skip format, input
03 OC AC 17 E1 0451 575      getskip:bbc      #atr v string, fcb_l_attr(ap), 5$ ;if string i/o
FD57 31 0456 576      brw      invstrfmt      ;fail with invalid string format
003F 30 0459 577 5$:      bsbw      get_format_parm_1      ;get the number of records to skip
03 18 045C 578      bgeq      10$      ;if geq, ok
FD45 31 045E 579      brw      invfrmpm      ;its leq, invalid format
52 51 D0 0461 580 10$:      movl    r1,r2      ;set number to skip
00000000'GF 16 0464 581      jsb      g^pliss$getskip_r2      ;skip em
FBB1 31 046A 582      brw      pliss$getfmt_r6      ;go again
046D 583
046D 584      ; x format, input
002B 30 046D 585      getx:      bsbw      get_format_parm_1      ;get the number of chars to skip
09 13 0470 586      beql      10$      ;if eql, ignore it
50 51 D0 0472 587      movl    r1,r0      ;set width
00000000'GF 16 0475 588      jsb      g^pliss$getnedi_r6      ;get that number of chars
FBA0 31 047B 589 10$:      brw      pliss$getfmt_r6      ;go again
047E 590
047E 591      ; right paren - end of iteration. the iteration count on the format stack is
047E 592      ; decremented. if it is <= 0, we go on to the next format item. otherwise, we
047E 593      ; repeat the iterated items.
047E 594      getrparen:
FB9C CF 9F 047E 595      pushab   pliss$getfmt_r6      ;set return addr
06 11 0482 596      brb      comparen      ;cont in common
0484 597      putrparen:
0000053C'EF 9F 0484 598      pushab   pliss$putfmt_r6      ;set return addr
048A 599      comparen:
52 6B D0 048A 600      movl    str_l_sp(r11),r2      ;get format sp
05 62 F4 048D 601      sobgeq   (r2),T0$      ;do an iteration
6B 08 A2 9E 0490 602      movab    8(r2),str_l_sp(r11)      ;clean stack
05 0494 603      rsb                    ;process next format item
04 AB 04 A2 D0 0495 604 10$:      movl    4(r2),str_l_fp(r11)      ;restart this format
05 049A 605      rsb                    ;process next format item
049B 606
049B 607      ;get_format_parm_1 - get a format parm. if the parm is missing, 1 is supplied
049B 608      ; as default.
049B 609      ; inputs:
049B 610      ;      r11 - address of stream block
049B 611      ;      ap - address of fcb
049B 612      ; outputs:
049B 613      ;      r1 - value of parameter or 1 if item missing
049B 614      ;
049B 615      get_format_parm_1:
51 01 D0 049B 616      movl    #1,r1      ;set missing parm value
02 11 049E 617      brb      get_format_com      ;cont in common
04A0 618
04A0 619      ;get_format_parm - get a format parm. if the parm is missing, 0 is supplied
04A0 620      ; as default.
04A0 621      ; inputs:
04A0 622      ;      r11 - address of stream block
04A0 623      ;      ap - address of fcb
04A0 624      ; outputs:
04A0 625      ;      r1 - value of parameter or 0 if item missing
04A0 626      ;
04A0 627      get_format_parm:
```

```
51 D4 04A0 628      cirl r1 ;set missing parm value
OC AB 01 CA 04A2 629 get_format_com:
50 04 BB 90 04A6 630      bicl #str_m_missing,str_l_fs(r11) ;clear missing parameter
04 AB D6 04AA 631 ; get the parameter type and case on it
04 AD 632      movb @str_l_fp(r11),r0 ;get parm type
04 AD 633      incl str_l_fp(r11) ;increment format pointer
04 AD 634      case type=5,r0,< - ;case on parameter type
04 AD 635      10$, - ;missing
04 AD 636      20$, - ;byte constant
04 AD 637      30$, - ;word constant
04 AD 638      40$, - ;long constant
04 AD 639      50$, - ;pc relative long constant
04 AD 640      60$, - ;pc relative long entry point (V1)
04 AD 641      70$, - ;invalid format
04 AD 642      70$> ;pc relative long entry point (V2)
FCDB 31 04C1 643 5$: brw invfrm ;none of the above, invalid format
OC AB 01 C8 04C4 644 ; missing
51 D5 04C8 645 10$: bisl #str_m_missing,str_l_fs(r11) ;set missing parameter
05 04CA 646      tstl r1 ;set default value condition code
04 CB 647      rsb ;return
51 04 BB 98 04CB 648 ; byte constant
04 AB D6 04CF 649 20$: cvtbl @str_l_fp(r11),r1 ;get the parm
51 D5 04D2 650      incl str_l_fp(r11) ;increment format pointer
05 04D4 651      tstl r1 ;set cond codes
04 D5 652      rsb ;return
51 04 BB 32 04D5 653 ; word constant
04 AB 02 C0 04D9 654 30$: cvtl @str_l_fp(r11),r1 ;get the parm
51 D5 04DD 655      addl #2,str_l_fp(r11) ;increment format pointer
05 04DF 656      tstl r1 ;set cond codes
04 E0 657      rsb ;return
51 04 BB D0 04E0 658 ; long constant
04 AB 04 C0 04E4 659 40$: movl @str_l_fp(r11),r1 ;get the parm
51 D5 04E8 660      addl #4,str_l_fp(r11) ;increment format pointer
05 04EA 661      tstl r1 ;set cond codes
04 EB 662      rsb ;return
51 04 BB D0 04EB 663 ; pc relative long constant, used for remote and picture formats
04 AB 04 C0 04EF 664 50$: movl @str_l_fp(r11),r1 ;get the parm
51 04 AB C0 04F3 665      addl #4,str_l_fp(r11) ;increment format pointer
05 04F7 666      addl str_l_fp(r11),r1 ;make addr absolute
04 F8 667      rsb ;return
50 04 BB D0 04F8 668 ; version 1 pc relative entry point, used for expressions in format items
04 AB 04 C0 04FC 669 60$: movl @str_l_fp(r11),r0 ;get the addr
50 04 AB C0 0500 670      addl #4,str_l_fp(r11) ;update format pointer
51 FC AD D0 0504 671      addl str_l_fp(r11),r0 ;make addr absolute
60 00 FB 0508 672      movl -4(fp),r1 ;set parent frame pointer
51 50 D0 050B 673      calls #0,(r0) ;call it
05 050E 674      movl r0,r1 ;set parm
050F 675      rsb ;return
050F 676
050F 677 ; version 2 pc relative entry point, used for expressions in format items
050F 678 70$:
50 04 BB D0 050F 679      movl @str_l_fp(r11),r0 ;get the addr
04 AB 04 C0 0513 680      addl #4,str_l_fp(r11) ;update format pointer
50 04 AB C0 0517 681      addl str_l_fp(r11),r0 ;make addr absolute
51 08 AB D0 051B 682      movl str_l_parent(r11),r1 ;set parent frame pointer
60 00 FB 051F 683      calls #0,(r0) ;call it
51 50 D0 0522 684      movl r0,r1 ;set parm
```



```
05 0525 685          rsb                      ;return
      0526 686
      0526 687 fail:  movl    r0,fcbl_error(ap)    ;set error in fcb
      052A 688          pushl   ap                ;set fcb addr
      052C 689          pushl   r0                ;set error code
      052E 690          pushl   #pli$error         ;set error condition
      0534 691          calls   #3,g^plisio_error ;signal error
      053B 692          ret                      ;return
      053C 693
      053C 694      ++
      053C 695      pli$$putfmt_r6
      053C 696
      053C 697      functional description:
      053C 698          control formats are processed and the next item is transmitted to the
      053C 699          file buffer via edit directed output. for data formats, the general
      053C 700          flow is: the compiled code jsb's to pli$pute*** routine. that
      053C 701          routine pushes the address, scale and precision and the case index
      053C 702          for the general conversion routine for the data type of the source.
      053C 703          for unaligned bit targets, the offset is passed in r5. thus r5 MUST
      053C 704          BE PRESERVED by all output control formats or the offset is lost.
      053C 705          the pli$pute*** routine then jmp's to this routine. this routine
      053C 706          finds the next data format (processing all intervening control formats)
      053C 707          and then enters the data format processing code. the data formats
      053C 708          convert the source to a standard type based on the format. this is then
      053C 709          placed in the files buffer by jumping to pli$$putnedi_r6.
      053C 710          note that some of the common control formats are above in the getformat
      053C 711          section.
      053C 712      inputs:
      053C 713          0(sp) - data type as a case index for pli$cvrt_cg_r3 as source
      053C 714          4(sp) - address of next item to put
      053C 715          8(sp) - prec/scale of item
      053C 716          12(sp) - return address
      053C 717          r11 - address of stream block
      053C 718          ap - address of fcb
      053C 719      outputs:
      053C 720      side effects:
      053C 721          r0-r6 are destroyed
      053C 722      --
      053C 723
      053C 724      pli$$putfmt_r6::
      053C 725          movb    astr_l fp(r11),r0      ;get format type
      0540 726          incl    str_l fp(r11)        ;update format pointer
      0543 727          case    type=B,r0,limit=#1,< - ;case on format type
      0543 728              putbiter, -                ;1 byte constant iteration
      0543 729              putwiter, -                ;2 word constant iteration
      0543 730              putliter, -                ;3 long constant iteration
      0543 731              invfrm, -                  ;4 pc relative iter (invalid)
      0543 732              putexpriter, -            ;5 expression iteration (Version 1)
      0543 733              puteof, -                ;6 end of format
      0543 734              putexpriter_v2, -        ;7 expression iteration (Version 2)
      0543 735              invfrm, -                ;8 invalid format
      0543 736              invfrm, -                ;9 invalid format
      0543 737              puta, -                    ;10 alphanumeric format
      0543 738              putb1, -                  ;11 bit (1) format
      0543 739              putb1, -                  ;12 bit 1 format
      0543 740              putb2, -                  ;13 bit 2 format
      0543 741              putb3, -                  ;14 bit 3 format
```

```
0543 742 putb4, - ;15 bit 4 format
0543 743 putcol, - ;16 column format
0543 744 putcol, - ;17 column format
0543 745 pute, - ;18 exp format
0543 746 putf, - ;19 fixed format
0543 747 putline, - ;20 line format
0543 748 putp, - ;21 picture format
0543 749 putpage, - ;22 page format
0543 750 putr, - ;23 remote format (PL/I version 1)
0543 751 putskip, - ;24 skip format
0543 752 puttab, - ;25 tab format
0543 753 putx, - ;26 blank format
0543 754 invfrm, - ;27 left paren (no longer used)
0543 755 putrparen, - ;28 right paren
0543 756 putr_v2> ;29 remote format (PL/I version 2)
0581 757
FC18 31 0581 758 brw invfrm ;none of the above, invalid format
0584 759
0584 760 ; a format, output
0584 761 ; get the width
FF19 30 0584 762 puta: bsbw get_format_parm ;get the parameter
1F 0C AB 00 E1 0587 763 bbc #str_v_missing,str_l,fs(r11),20$ ;if parm missing then[
53 000003E8 8F D0 058C 764 ; if the width is missing, we convert the source to a char(1000) var.
52 18 AB 9E 0593 765 movl #1000,r3 ;set max size for vcha in field
54 8E 06 C1 0597 766 movab str_b_field(r11),r2 ;set addr
50 50 8E 7D 059B 767 addl3 #cvt_k_dst_vcha,(sp)+,r4 ;set case index for vcha dest
00000000'GF 00 FB 059E 768 10$: movq (sp)+,r0 ;set src addr, and prec
05A5 769 calls #0,g^pli$cvrt_cg_r3 ;convert src to vcha
00000000'GF 17 05A5 770 ; put it out
05AB 771 jmp g^pli$putnedi_r6 ;put it in buffer and return]
51 D5 05AB 772 ;width present
29 13 05AD 773 20$: tstl r1 ;else [set cond codes
03 14 05AF 774 beql 50$ ;if eql, ignore this field
000003E8 8F 51 D1 05B1 775 bgtr 30$ ;if gtr, cont
FFB2 31 05B1 776 brw invfrmprm ;its lss, invalid format
0A 15 05B8 777 30$: cmpl r1,#1000 ;len too big for field?
50 00000000'8F D0 05BD 778 bleq 40$ ;if leg, no
FF5F 31 05C4 779 movl #pli$_strovfl,r0 ;set field overflow
18 AB 51 B0 05C7 780 brw fail ;and fail
53 51 D0 05CB 781 40$: movw r1,str_b_field(r11) ;set len in field
52 1A AB 9E 05CE 782 movl r1,r3 ;set dst len
54 8E 05 C1 05D2 783 movab <str_b_field+2>(r11),r2 ;set dst addr
C3 11 05D6 784 addl3 #cvt_k_dst_char,(sp)+,r4 ;set case index for char dest
5E 0C AE 9E 05D8 785 10$ brb ;cont]
05 05DC 786 50$: movab 12(sp),sp ;clean stack
05DD 787 rsb ;its a(0), ignore field by returning
05DD 788
05DD 789 ; b format, output
05DD 790 ; set the radix
01 DD 05DD 791 putb1: pushl #1 ;set radix
0A 11 05DF 792 brb putb ;cont in common
02 DD 05E1 793 putb2: pushl #2 ;set radix
06 11 05E3 794 brb putb ;cont in common
03 DD 05E5 795 putb3: pushl #3 ;set radix
02 11 05E7 796 brb putb ;cont in common
04 DD 05E9 797 putb4: pushl #4 ;set radix
05EB 798 ;stack at this point:
```



```
05EB 799      :12(sp) prec/scale of src
05EB 800      : 8(sp) addr of src
05EB 801      : 4(sp) datatype of src
05EB 802      : 0(sp) radix
      FEB2 30 05EB 803 putb: bsbw get_format_parm ;get the width
      OF 14 05EE 804      bgtr 20$ ;if gtr, cont
      03 13 05F0 805      beql 10$ ;if eql, check for missing
      FBB1 31 05F2 806      brw invfrmprm ;its lss, invalid format
05 OC AB 00 E0 05F5 807 10$: bbs #str_v_missing,str_l_fs(r11),20$ ;if parm missing, use src prec
SE 10 AE 9E 05FA 808      movab 16(sp),sp ;its a(0), so clean stack
      05 05FE 809      rsb ;return
      18 AB SE D0 05FF 810 ; determine the binary precision of the src
50 04 AE 09 C7 0603 811 20$: movl sp,str_b_field(r11) ;save current stack addr
      0606 812      movl r1,r6 ;save width
      060B 813      divl3 #9,4(sp),r0 ;get data type of source
      060B 814      case type=b,r0,< - ;case on src data type
      060B 815      35$, - ;0 pic, not yet implemented
      060B 816      30$, - ;1 fixb
      060B 817      30$, - ;2 fltb
      060B 818      40$, - ;3 fixd
      060B 819      40$, - ;4 fltd
      060B 820      30$, - ;5 char
      060B 821      60$, - ;6 vcha
      060B 822      30$, - ;7 bit
      060B 823      30$> ;8 abnt
      53 53 OC AE D0 0621 824      brw invfrm ;invalid data type, fail
53 00000080 8F CA 0624 825 30$: movl 12(sp),r3 ;set dst prec eql to src prec
      58 11 062F 826      bicl #^x80,r3 ;clear gfloat indicator
      51 OC AE D0 0631 827 35$: brb 70$ ;cont
      53 61 9A 0635 828      movl 12(sp),r1 ;get addr of pic descr
      51 01 A1 9A 0638 829      movzbl pic$w_pq(r1),r3 ;get src prec
      11 11 063C 830      movzbl pic$w_pq+1(r1),r1 ;get src scale
      53 OC AE 9A 063E 831      brb 45$ ;cont
53 00000080 8F CA 0642 832 40$: movzbl 12(sp),r3 ;get src prec
51 OC AE F8 8F 78 0649 833      bicl #^x80,r3 ;clear gfloat indicator
      53 51 C2 064F 834 45$: ashl #-8,12(sp),r1 ;get src scale
      1F 15 0652 835      subl r1,r3 ;get number of digs in integer part
53 0000014C 8F C4 0654 836      bleq 50$ ;if leq, then result is zero
53 00000063 8F C0 065B 837      mull2 #332,r3 ;get binary precision according to rule
53 00000064 8F C6 0662 838      addl #99,r3 ;round for ceil and fixed divide
      1F 53 D1 0669 839      divl2 #100,r3 ;finish (r3=ceil((p-q)*3.32)) really!
      1B 15 066C 840      cmpl r3,#31 ;prec > 31?
      53 1F D0 066E 841      bleq 70$ ;if leq, no, continue
      16 11 0671 842      movl #31,r3 ;use max fixb prec
      51 18 AB 9E 0673 843      brb 70$ ;cont
61 56 20 81 56 B0 0677 844 50$: movab str_b_field(r11),r1 ;get addr of field
      SE 10 AE 9E 067A 845      movw r6,(r1)+ ;put in width
      53 08 BE 3C 0684 846      movc5 #0,(sp),#^x20,r6,(r1) ;blank it out
      50 6E D0 0689 847      movab 16(sp),sp ;clean stack
      53 50 C0 068C 848      rsb ;return
      53 50 D7 068F 849 60$: movzwl @8(sp),r3 ;get cur len of vcha src
      53 50 C6 0691 850 ;get size of bit temp needed, based on src prec and radix
      53 50 C4 0694 851 70$: movl (sp),r0 ;get radix
      53 50 C4 0694 852      addl r0,r3 ;round prec up to next multiple of radix
      53 50 C4 0694 853      decl r3 ;
      53 50 C4 0694 854      divl r0,r3 ;
      53 50 C4 0694 855      mull r0,r3 ;
```

```
50 50 53 07 C1 0697 856      addl3 #7,r3,r0      ;round prec up to a byte
50 50 FD 8F 78 0698 857      ashl #4,r0,r0      ;get number of bytes required
      D1 13 06A0 858      beql 50$      ;if eql, then result is 0
      06A2 859 ;allocate temp on stack and clear last byte
      SE 50 C2 06A2 860      subl r0,sp      ;get space for temp on stack
      FF AE40 94 06A5 861      clrb -1(sp)[r0]      ;clear last byte of temp
      52 SE D0 06A9 862      movl sp,r2      ;set addr of temp
      54 18 AB D0 06AC 863      movl str_b_field(r11),r4      ;get old stack pointer
      50 08 A4 7D 06B0 864      movq 8(r4),r0      ;get original src
      08 A4 53 D0 06B4 865      movl r3,8(r4)      ;save number of bits in temp
      0C A4 56 D0 06B8 866      movl r6,12(r4)      ;save width of field
      05 12 06BC 867      bneq 80$      ;if neq, cont
      0C A4 53 64 C7 06BE 868      divl3 (r4),r3,12(r4)      ;use converted prec for missing width
      54 04 A4 08 C1 06C3 869 80$: addl3 #cvt_k_dst_abt,4(r4),r4 ;set case index for abt dst
00000000'GF 00 FB 06C8 870      calls #0,pli$cvrt_cg,r3 ;convert src to abt temp
      06CF 871 ;convert abt temp to vcha in field using B-radix conversion
      06CF 872 ;local register usage for conversion:
      06CF 873 ; r0 - radix
      06CF 874 ; r1 - address of table for this radix
      06CF 875 ; r2 - current position in bit string
      06CF 876 ; r3 - output pointer
      06CF 877 ; r4 - current bits or char
      06CF 878 ; r5 - number of chars left to do
      06CF 879 ; r6 - requested width, number of blanks to append
      54 18 AB D0 06CF 880      movl str_b_field(r11),r4      ;get old stack pointer
      50 50 64 D0 06D3 881      movl (r4),r0      ;get radix
      51 08 A4 D0 06D6 882      movl 8(r4),r1      ;get number of bits in temp
      56 0C A4 D0 06DA 883      movl 12(r4),r6      ;get req width
000003E8 8F 56 D1 06DE 884      cmpl r6,#1000      ;width too big?
      0A 15 06E5 885      bleq 90$      ;if leg, no
50 00000000'8F D0 06E7 886      movl #pli$_strovfl,r0      ;set field overflow
      FE35 31 06EE 887      brw fail      ;and fail
      53 18 AB 9E 06F1 888 90$: movab str_b_field(r11),r3      ;get addr of start of output field
      83 56 B0 06F5 889      movw r6,(r3)+      ;set length in field
      55 51 50 C7 06F8 890      divl3 r0,r1,r5      ;get number of bytes of output
      0C AB 08 CA 06FC 891      bicl #str_m_blankend,str_l_fs(r11) ;assume we can fill req. width
      56 55 D1 0700 892      cmpl r5,r6      ;enough to fill requested width?
      07 19 0703 893      blss 100$      ;if lss, no
      0705 894      ;if gtr and stringsize supported
      0705 895      ;then raise it here
      55 56 D0 0705 896      movl r6,r5      ;set req width as length
      56 56 D4 0708 897      clrl r6      ;set no blanks on end
      07 11 070A 898      brb 110$      ;cont
      0C AB 08 C8 070C 899 100$: bisl #str_m_blankend,str_l_fs(r11) ;remember to blank out end
      56 55 C2 0710 900      subl r5,r6      ;get number of blanks for end
      51 01 50 78 0713 901 110$: ashl r0,#1,r1      ;get table address
      51 F8E2 CF41 9E 0717 902      movab bformattab-2(pc)[r1],r1 ;based on radix
      52 D4 071D 903      clrl r2      ;start at beginning of bit string
      54 DD 071F 904      pushl r4      ;save old stack pointer
      0D 11 0721 905      brb 130$      ;enter loop
      54 04 AE 50 52 EF 0723 906 120$: extzv r2,r0,4(sp),r4      ;get some bits
      83 6144 90 0729 907      movb (r1)[r4],(r3)+      ;store resulting char in field
      52 50 C0 072D 908      addl r0,r2      ;update pos in bit string
      FO 55 F4 0730 909 130$: sobgeq r5,120$      ;go again
      0733 910 ; append blanks if necessary
      63 56 06 0C AB 03 E1 0733 911      bbc #str_v_blankend,str_l_fs(r11),140$ ;if we must append blanks
      20 6E 00 2C 0738 912      movc5 #0,(sp),#^x20,r6,(r3) ;append blanks
```



```
5E 8E 10 C1 073E 913 140$: addl3 #16,(sp)+,sp ;clean stack
00000000'GF 17 0742 914 jmp g^plissputnedi_r6 ;put it out
0748 915
0748 916 ; column format, output
0748 917 ; if the requested column is greater than current column and less than the
0748 918 ; linesize, we put in enough blanks to position to the requested column.
0748 919 ; if the requested column is greater than linesize we do a skip. if the
0748 920 ; requested column is less than current column, we do a skip and then
0748 921 ; fill with blanks to get to the requested column
03 OC AC 17 E1 0748 922 putcol: bbc #atr_v_string,fcbl_attr(ap),5$ ;if string i/o
FA60 31 074D 923 brw invstrfmt ;fail with invalid string format
FD4D 30 0750 924 5$: bsbw get_format_parm ;get the parameter
07 14 0753 925 bgtr 20$ ;if gtr, cont
03 13 0755 926 beql 10$ ;if eql, cont
FA4C 31 0757 927 brw invfrmpm ;parm < 0, invalid format
51 D6 075A 928 10$: incl r1 ;use 1 instead of 0
50 2A AC 3C 075C 929 20$: movzwl fcb_w_linesize(ap),r0 ;get linesize
50 51 B1 0760 930 cmpw r1,r0 ;req col > linesize
03 15 0763 931 bleq 30$ ;if leq, no, cont
51 01 D0 0765 932 movl #1,r1 ;use 1 for col
51 D7 0768 933 30$: decl r1 ;get req col - 1
2E AC 51 B1 076A 934 cmpw r1,fcbl_w_column(ap) ;(requested col-1) > current col?
12 14 076E 935 bgtr 50$ ;if gtr, then yes
0D 13 0770 936 beql 40$ ;if eql, then already at right col
51 DD 0772 937 pushl r1 ;save req col
00000000'GF 16 0774 938 jsb g^plissputskp1_r2 ;do a skip
51 8ED0 077A 939 popl r1 ;restore req col
07 14 077D 940 bgtr 60$ ;if eql, just return
51 FD8A 31 077F 941 40$: brw plissputfmt_r6 ;go again
51 2E AC A2 0782 942 50$: subw fcb_w_column(ap),r1 ;get number of blanks to move
032E 31 0786 943 60$: brw blank_field ;fill with blanks, put in buf, go again
0789 944
0789 945 ; e format, output
0789 946 ;get prec of float dec temp from src dtyp and prec
50 OC AB 10 CA 0789 947 pute: bicl #str_m_gfloat,src_l_fs(r1) ;assume not g float src
50 6E 09 C7 078D 948 divl3 #9,(sp),r0 ;get data type of source
0791 949 case type=b,r0,< - ;case on data type
0791 950 5$, - ;0 pic
0791 951 10$, - ;1 fixb
0791 952 10$, - ;2 fltb
0791 953 30$, - ;3 fixd
0791 954 30$, - ;4 fltd
0791 955 50$, - ;5 char
0791 956 45$, - ;6 vcha
0791 957 40$, - ;7 bit
0791 958 40$> ;8 abit
51 F9F2 31 07A7 959 brw invfrm ;invalid data type, fail
51 08 AE D0 07AA 960 5$: movl 8(sp),r1 ;get addr of pic descr
53 53 61 9A 07AE 961 movzbl pic$w_pq(r1),r3 ;get prec of pic src
53 2E 11 07B1 962 brb 35$ ;cont
53 08 AE D0 07B3 963 10$: movl 8(sp),r3 ;get prec of binary src
04 53 07 E5 07B7 964 bbcc #7,r3,20$ ;if g float
OC AB 10 C8 07BB 965 bicl #str_m_gfloat,src_l_fs(r1) ;set gfloat
53 00000064 8F C4 07BF 966 20$: mull #100,r3 ;get pl1 decimal prec
53 0000014B 8F C0 07C6 967 addl #331,r3 ;
53 0000014C 8F C6 07CD 968 divl #332,r3 ;
12 6E 91 07D4 969 cmpb (sp),#cvt_k_src_fltb ;float bin src?
```

```
2B 12 07D7 970 bneq 60$ ;if neq, no, cont
53 53 D7 07D9 971 decl r3 ;correct prec for context computation
27 11 07DB 972 brb 60$ ;cont
53 08 AE 9A 07DD 973 30$: movzbl 8(sp),r3 ;get prec of decimal src
1F 53 07 E5 07E1 974 35$: bbcc #7,r3,60$ ;if g float
OC AB 10 C8 07E5 975 bisl #str_m_gfloat,str_l_fs(r11) ;set gfloat
19 11 07E9 976 brb 60$ ;cont
53 1F D0 07EB 977 40$: movl #31,r3 ;use max fixb prec for bit
CF 11 07EE 978 brb 20$ ;cont
50 04 AE D0 07F0 979 45$: movl 4(sp),r0 ;get addr of string
51 80 3C 07F4 980 movzwl (r0)+,r1 ;and size (point past 1st word)
08 11 07F7 981 brb 55$
51 08 AE 9A 07F9 982 50$: movzbl 8(sp),r1 ;get size of src
50 04 AE D0 07FD 983 movl 4(sp),r0 ;get addr of src
02DF 30 0801 984 55$: bsbw charflt_ctx ;get flt dec context
53 DD 0804 985 60$: pushl r3 ;save dec prec
0806 986 ; get context of fltb temp
OB OC AB 04 E1 0806 987 bbc #str_v_gfloat,str_l_fs(r11),80$ ;if g float src
01 DD 080B 988 pushl #1 ;set for g context
53 00000080 8F C8 080D 989 bisl #128,r3 ;set g float bit for convert
0E 11 0814 990 brb 100$ ;cont
53 0F D1 0816 991 80$: cmpl #15,r3 ;is it f or d?
07 19 0819 992 blss 90$ ;if lss, no
7E D4 081B 993 clrl -(sp) ;set for d context
53 0F D0 081D 994 movl #15,r3 ;set max prec of d
02 11 0820 995 brb 100$ ;cont
02 DD 0822 996 90$: pushl #2 ;set for h context
12 08 AE 91 0824 997 100$: cmpb 8(sp),#cvt_k_src_fltb ;float bin src?
03 12 0828 998 bneq 105$ ;if neq, no, cont
04 AE D6 082A 999 incl 4(sp) ;correct dec prec
082D 1000 ;allocate fltb temp on stack
5E 10 C2 082D 1001 105$: subl #16,sp ;get room for temp
52 5E D0 0830 1002 movl sp,r2 ;set temp addr for dst
50 1C AE 7D 0833 1003 movq 28(sp),r0 ;set src addr and prec
54 18 AE 04 C1 0837 1004 addl3 #cvt_k_dst_fltb,24(sp),r4 ;set convert index, dst = fltb
083C 1005 ; convert src to fltb
00000000'GF 00 FB 083C 1006 calls #0,g^pliscvrt_cg_r3 ;convert to fltb
0843 1007 ; get w,d,s. s is ignored
FC5A 30 0843 1008 bsbw get_format_parm ;get the width
000003E8 8F 51 D1 0846 1009 cmpl r1,#1000 ;too big?
0A 15 084D 1010 bleq 110$ ;if leq, no
50 00000000'8F D0 084F 1011 movl #pl1$_strovfl,r0 ;set field overflow
FCCD 31 0856 1012 brw fail ;and fail
56 51 D0 0859 1013 110$: movl r1,r6 ;save it
FC41 30 085C 1014 bsbw get_format_parm ;get the digs in frac
OB OC AB 00 E0 085F 1015 bbs #str_v_missing,str_l_fs(r11),130$ ;if digs in frac not missing
53 51 D0 0864 1016 movl r1,r3 ;save digs in frac
19 18 0867 1017 bgeq 140$ ;if geq, cont
F93A 31 0869 1018 brw invfrprm ;set invalid format
53 14 AE 01 C3 086C 1019 130$: subl3 #1,20(sp),r3 ;use dec prec of src-1 as digs in frac
50 10 AE 07 C1 0871 1020 addl3 #7,16(sp),r0 ;get number of chars for exp,sign, dot
50 56 50 C3 0876 1021 subl3 r0,r6,r0 ;get max number of digs in frac
50 53 D1 087A 1022 cmpl r3,r0 ;src-1 digs too many?
03 15 087D 1023 bleq 140$ ;if leq, no, use src-1
53 50 D0 087F 1024 movl r0,r3 ;use number of digs in frac that fits
FC1B 30 0882 1025 140$: bsbw get_format_parm ;get scale but ignore it
0885 1026 ; set up parms for convert routine
```


PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419	Op420	Op421	Op422	Op423	Op424	Op425	Op426	Op427	Op428	Op429	Op430	Op431	Op432	Op433	Op434	Op435	Op436	Op437	Op438	Op439	Op440	Op441	Op442	Op443	Op444	Op445	Op446	Op447	Op448	Op449	Op450	Op451	Op452	Op453	Op454	Op455	Op456	Op457	Op458	Op459	Op460	Op461	Op462	Op463	Op464	Op465	
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--

				092A	1084	:	movq	12(sp),r0	;set src addr and prec
				092A	1085	:	movl	(sp),r3	;get digs in frac
				092A	1086	:	cmpl	#31,(sp)	;trying to print more than 31 digs?
	6E	1F	D1	092D	1087	:	bgeq	60\$;if geq, no
		03	18	092F	1088	50\$:	brw	invfrmprm	;invalid format
		F874	31	0932	1089	60\$:	brb	90\$;go output rounded format
		36	11	0934	1090	:	ashl	#8,r3,r3	;use digs in frac as scale
				0934	1091	:	movb	#31,r3	;use max fixd prec
				0934	1092	:	subl	#16,sp	;get space for fixd temp
				0934	1093	:	movl	sp,r2	;set tmp addr
				0934	1094	:	calls	#0,g^plisfltbfixed_r6	;convert fltb to fixd
				0934	1095	:	brb	110\$;cont
	51	10	AE	0934	1096	65\$:	movl	16(sp),r1	;get addr of picture descr
		51	61	0938	1097	:	movzwl	pic\$w,pq(r1),r1	;get prec and scale
		51	1F	093B	1098	:	cmpb	#31,r1	;prec >= 31?
			04	093E	1099	:	brb	75\$;cont
				0940	1100	70\$:	;decimal		
	10	AE	1F	0940	1101	:	cmpb	#31,16(sp)	;prec >= 31?
			24	0944	1102	75\$:	bgtr	90\$;if gtr, no, use common
			E7	0946	1103	:	blss	50\$;if lss, invalid src prec
		OC	AE	0948	1104	:	movq	12(sp),r0	;set src addr and prec
		53	6E	094C	1105	:	movl	(sp),r3	;get digs in frac
		1F	53	094F	1106	:	cmpl	r3,#31	;trying to print more than 31 digs?
			DB	0952	1107	:	bgtr	50\$;if gtr, then yes, invalid format
	53	53	08	0954	1108	:	ashl	#8,r3,r3	;use digs in frac as scale
		53	1F	0958	1109	:	movb	#31,r3	;use max fixd prec
		5E	10	095B	1110	:	subl	#16,sp	;get room for fixd temp
		52	5E	095E	1111	:	movl	sp,r2	;set addr of tmp
	00000000	'GF	00	0961	1112	:	calls	#0,g^plisfixdfixed_r6	;convert fixd to fixd tmp
			3D	0968	1113	:	brb	110\$;cont
		50	OC	096A	1114	90\$:	movq	12(sp),r0	;set src addr and prec
		03	08	096E	1115	:	addl3	8(sp),#cvt_k_dst_fixd,r4	;set case index
	54			0973	1116	:	addl3	(sp),#1,r3	;get digs in frac + 1
		53	01	0977	1117	:	cmpl	r3,#31	;trying to print more than 31 digits?
			1F	097A	1118	:	bleq	100\$;if leg, no
			03	097C	1119	:	brw	invfrmprm	;invalid format
			F827	097F	1120	100\$:	ashl	#8,r3,r3	;use digs in frac + 1 as scale
	53	53	08	0983	1121	:	movb	#31,r3	;set max fixd prec
		53	1F	0986	1122	:	subl	#16,sp	;get room for fixd temp
		5E	10	0989	1123	:	movl	sp,r2	;set addr of tmp
		52	5E	098C	1124	:	calls	#0,g^pliscvrt_cg_r3	;convert src to fixd
	00000000	'GF	00	0993	1125	:	subl	#16,sp	;get room for another temp
		5E	10	0996	1126	:	ashp	#-1,#31,16(sp),#5,#31,(sp)	;round temp
1F	05	10	AE	099E		:			
			1F	099F	1127	:	movq	(sp)+,8(sp)	;copy to orig temp
			FF	09A3	1128	:	movq	(sp)+,8(sp)	;cont
		08	AE	09A7	1129	110\$:	;at this point stack looks like:		
		08	AE	09A7	1130	:	; 0(sp) - rounded fixd(31,digs in frac) temp		
				09A7	1131	:	; 16(sp) - digs in frac		
				09A7	1132	:	; 20(sp) - width		
				09A7	1133	:	; 24(sp) - src data type		
				09A7	1134	:	; 28(sp) - src addr		
				09A7	1135	:	; 32(sp) - src prec		
				09A7	1136	:	; 36(sp) - return addr		
		50	SE	09A7	1137	:	movl	sp,r0	;set addr of fixd temp src
51	10	AE	08	09AA	1138	:	ashl	#8,16(sp),r1	;use digs in frac as scale
		51	1F	09AF	1139	:	movb	#31,r1	;use 31 as prec of fixd src


```

      5E 22 C2 09B2 1140      subl #34,sp      ;get space for char temp
      52 22 D0 09B5 1141      movl sp,r2      ;set char temp addr of dst
      53 22 D0 09B8 1142      movl #34,r3      ;set 34 as len
00000000'GF 00 FB 09BB 1143      calls #0,g^pli$fixdchar_r6 ;convert fixd to char
      56 36 AE D0 09C2 1144      movl 54(sp),r6 ;get width
      18 AB 56 B0 09C6 1145      movw r6,str_b_field(r11) ;set width in field
      55 22 D1 09CA 1146      cmpl #34,r6      ;width < 34?
      1C 19 09CD 1147      blss 140$,      ;if lss, no
      54 22 56 C3 09CF 1148      subl3 r6,#34,r4 ;get number of leading blanks
      6E 54 20 3B 09D3 1149      skpc #^x20,r4,(sp) ;skip leading blanks
      03 13 09D7 1150      beql 120$      ;if eql, cont
      F7CA 31 09D9 1151      brw invfrmpm ;and fail
1A AB 61 56 28 09DC 1152 120$: movc3 r6,(r1),<str_b_field+2>(r11) ;copy result to field
      5E 46 AE 9E 09E1 1153 130$: movab 70(sp),sp ;clean stack
00000000'GF 17 C3 09E5 1154      jmp g^pli$putnedi_r6 ;put it out
      51 56 22 17 09EB 1155 140$: subl3 #34,r6,r1 ;get number of blanks needed
1A AB 51 20 6E 00 2C 09EF 1156      movc5 #0,(sp),#^x20,r1,<str_b_field+2>(r11) ;put in leading blanks
      63 6E 22 28 09F6 1157      movc3 #34,(sp),(r3) ;copy the result to field
      E5 11 09FA 1158      brb 130$      ;cont
      09FC 1159
      09FC 1160 ; line format
      03 OC AC 17 E1 09FC 1161 putline:bbc #atr_v_string,fcbl_attr(ap),5$ ;if string i/o
      F7AC 31 0A01 1162      brw invstrfmt ;fail with invalid string format
      0A OC AC 07 E0 0A04 1163 5$: bbs #atr_v_print,fcbl_attr(ap),10$ ;if print, cont
50 00000000'8F D0 0A09 1164      movl #pli$_notprint,r0 ;set not print file
      FB13 31 0A10 1165      brw fail ;and fail
      FABA 30 0A13 1166 10$: bsbw get_format_parm ;get the parm
00000000'GF 16 0A16 1167      jsb g^pli$putline_r6 ;process the line
      FB1D 31 0A1C 1168      brw pli$putfmt_r6 ;go again
      0A1F 1169
      0A1F 1170 ; p format output
      FA7E 30 0A1F 1171 putp: bsbw get_format_parm ;get the pict desc
      03 12 0A22 1172      bneq 10$      ;if neq, cont
      F775 31 0A24 1173      brw invfrm ;fail
      52 18 AB 9E 0A27 1174 10$: movab str_b_field(r11),r2 ;set dst addr
      53 51 D0 0A2B 1175      movl r1,r3 ;set addr of pict desc
      82 04 A1 9B 0A2E 1176      movzbw pic$b_byte_size(r1),(r2)+ ;set size of resulting string
      54 8E 00 C1 0A32 1177      addl3 #cvt_k_dst_pic,(sp)+,r4 ;set data type
      50 8E 7D 0A36 1178      movq (sp)+,r0 ;set addr, size of src
00000000'GF 00 FB 0A39 1179      calls #0,g^pli$cvrt_cg_r3 ;convert to pic
00000000'GF 17 0A40 1180      jmp g^pli$putnedi_r6 ;put it out
      0A46 1181
      0A46 1182 ; page format
      03 OC AC 17 E1 0A46 1183 putpage:bbc #atr_v_string,fcbl_attr(ap),5$ ;if string i/o
      F762 31 0A4B 1184      brw invstrfmt ;fail with invalid string format
      0A OC AC 07 E0 0A4E 1185 5$: bbs #atr_v_print,fcbl_attr(ap),10$ ;if print, cont
50 00000000'8F D0 0A53 1186      movl #pli$_notprint,r0 ;set not print file
      FAC9 31 0A5A 1187      brw fail ;and fail
      00000000'GF 16 0A5D 1188 10$: jsb g^pli$putpage_r6 ;do a put page
      FAD6 31 0A63 1189      brw pli$putfmt_r6 ;go again
      0A66 1190
      0A66 1191 ; skip format, output
      03 OC AC 17 E1 0A66 1192 putskip:bbc #atr_v_string,fcbl_attr(ap),5$ ;if string i/o
      F742 31 0A6B 1193      brw invstrfmt ;fail with invalid string format
      FA2A 30 0A6E 1194 5$: bsbw get_format_parm_1 ;get the number to skip
      52 51 D0 0A71 1195      movl r1,r2 ;copy number to skip
00000000'GF 16 0A74 1196      jsb g^pli$putskip_r2 ;do the skips
```

```
FABF 31 0A7A 1197 brw pli$$putfmt_r6 ;go again
0A7D 1198
0A7D 1199 ; tab format
03 0C AC 17 E1 0A7D 1200 puttab: bbc #atr_v_string, fcb_l_attr(ap), 5$ ;if string i/o
F72B 31 0A82 1201 brw invstrfmt ;fail with invalid string format
FA13 30 0A85 1202 5$: bsbw get_format_parm_1 ;get the tab stop
05 14 0A88 1203 bgtr 10$ ;if gtr, cont
25 13 0A8A 1204 beql 30$ ;if eql, go again
F717 31 0A8C 1205 brw invfrmpm ;its lss, invalid format
50 2E AC 3C 0A8F 1206 10$: movzwl fcb_w_column(ap), r0 ;get current column
53 50 07 CB 0A93 1207 bicl3 #7, r0, r3 ;round down to last tab stop
52 51 03 78 0A97 1208 ashl #3, r1, r2 ;get number of blanks for req tabs
52 52 53 C0 0A9B 1209 addl r3, r2 ;get ending column
52 2A AC B1 0A9E 1210 cmpw fcb_w_linesize(ap), r2 ;past end of line?
07 19 0AA2 1211 blss 20$ ;if lss, yes, cont
51 52 50 C3 0AA4 1212 subl3 r0, r2, r1 ;get number of blanks needed
000C 31 0AA8 1213 brw blank_field ;output blanks and go again
00000000'GF 16 0AAB 1214 20$: jsb g^pli$$putskp1_r2 ;do a skip
FA88 31 0AB1 1215 30$: brw pli$$putfmt_r6 ;go again
0AB4 1216
0AB4 1217 ; x format, output
F9E4 30 0AB4 1218 putx: bsbw get_format_parm_1 ;get the number of blanks
0AB7 1219 : brw blank_field ;put out blanks and go again
0AB7 1220
0AB7 1221 :+
0AB7 1222 :blank_field
0AB7 1223 :this routine puts the specified number of blanks in to the field in vcha
0AB7 1224 :format. it then calls pli$$putnedi_r6 and jumps to pli$$putfmt_r6.
0AB7 1225 :inputs:
0AB7 1226 : r1 - number of blanks
0AB7 1227 : outputs:
0AB7 1228 : none
0AB7 1229 : side effects:
0AB7 1230 : r0-r4, r6 are destroyed
0AB7 1231 : r5 is preserved for the offset to bit sources
0AB7 1232 :-
0AB7 1233 blank_field:
000003E8 8F 55 DD 0AB7 1234 pushl r5 ;save r5 in case a bit src is pending
51 D1 0AB9 1235 cmpl r1, #1000 ;trying to put too many blanks in?
0A 15 0AC0 1236 bleq 10$ ;if leg, no
50 00000000'8F D0 0AC2 1237 movl #pli$_strovfl, r0 ;set field overflow
FASA 31 0AC9 1238 brw fail ;and fail
1A AB 51 18 AB 51 F7 0ACC 1239 10$: cvtlw r1, str_b_field(r11) ;set size of string
20 6E 00 2C 0AD0 1240 movc5 #0, (sp), #^x20, r1, str_b_field+2(r11) ;put in the blanks
00000000'GF 16 0AD7 1241 jsb g^pli$$putnedi_r6 ;output the field
55 8ED0 0ADD 1242 popl r5 ;restore r5
FA59 31 0AE0 1243 brw pli$$putfmt_r6 ;go on to next format
0AE3 1244
0AE3 1245 :+
0AE3 1246 :
0AE3 1247 : charflt_ctx
0AE3 1248 :
0AE3 1249 : finds the appropriate float decimal precision for a character
0AE3 1250 : string based on the number of digits in the mantissa and
0AE3 1251 : the value of the exponent.
0AE3 1252 :
0AE3 1253 : inputs:
```



```
0AE3 1254 :
0AE3 1255 :
0AE3 1256 :
0AE3 1257 :
0AE3 1258 :
0AE3 1259 :
0AE3 1260 :
0AE3 1261 :
0AE3 1262 :
0AE3 1263 :
0AE3 1264 :
0AE3 1265 :
0AE3 1266 :
05 OC AC 1A E1 0AE3 1267 char_flt ctx:
53 10 AE D0 0AE8 1268 bbc #atr_v_flttrg, fcb_l_attr(ap), 4$ ; if flt target
60 51 20 BB 0AEC 1269 movl 16(sp), r3 ; set fltb prec of target
54 01 D0 0AED 1270 4$: rsb ; return
7A 11 0AEF 1271 pushr #^m<r0, r1, r2, r4, r5> ; save regs
52 50 7D 0AF8 1272 skpc #32, r1, (r0) ; skip leading blanks
52 20 C2 0AF5 1273 bneq 5$ ; if string not blank, br
2B 63 91 0AF8 1274 movl #1, r4 ; else set prec of 1
04 12 0B07 1275 brb 100$
53 D6 0B09 1276 5$: movq r0, r2 ; save new addr and length from skip
52 D7 0B0A 1277 locc #32, r0, (r3) ; throw out trailing blanks too
63 52 2E 0B0B 1278 subl r0, r2 ; find the number of non-blank chars
02 13 0B0C 1279 cmpb (r3), #^a/+ ; check for a sign
54 50 C2 0B0D 1280 beql 10$ ; br if found
2F 14 0B0E 1281 cmpb (r3), #^a/- ; minus?
07 AE 20 90 0B0F 1282 bneq 20$ ; br if no sign
52 45 8F 0B10 1283 10$: incl r3 ; point past it
63 52 65 8F 0B11 1284 decl r2
54 50 C2 0B12 1285 20$: movl r2, r4 ; make char. count the digit count
07 AE 20 90 0B13 1286 locc #^a/./, r2, (r3) ; check for decimal point
52 45 8F 0B14 1287 beql 30$ ; br if none
63 52 65 8F 0B15 1288 decl r4 ; deduct dec. pt. from digit count
07 AE 20 90 0B16 1289 30$: locc #^a/E/, r2, (r3) ; look for E
54 50 C2 0B17 1290 bneq 40$ ; br if found
07 AE 20 90 0B18 1291 locc #^a/e/, r2, (r3) ; e?
52 45 8F 0B19 1292 beql 100$ ; if none, that's it
63 52 65 8F 0B20 1293 40$: subl r0, r4 ; sub. exponent chars from digit count
07 AE 20 90 0B21 1294 incl r1 ; point past the E/e
54 50 C2 0B22 1295 decl r0
2B 63 91 0B23 1296 cmpb (r1), #^a/+ ; check for exponent sign
05 13 0B24 1297 beql 45$ ; br if found
2D 61 91 0B25 1298 cmpb (r1), #^a/- ; minus?
04 12 0B26 1299 bneq 50$ ; br if no sign
51 D6 0B27 1300 45$: incl r1 ; point past the sign char
0F 54 D1 0B28 1301 decl r0
52 45 8F 0B29 1302 50$: cmpl r4, #15 ; is prec. huge?
07 AE 20 90 0B30 1303 bgtr 100$ ; if so, that's it
54 50 C2 0B31 1304 ; else, get exponent value
07 AE 20 90 0B32 1305 subl r0, sp ; get a stack temp
08 AE 61 50 BB 0B33 1306 pushr #^m<r0, r4> ; save some regs
11 BA 0B34 1307 movc3 r0, (r1), 8(sp) ; copy exp. digits to temp
07 AE 20 90 0B35 1308 popr #^m<r0, r4> ; restore regs
07 AE 20 90 0B36 1309 clrq -(sp) ; more temps
07 AE 20 90 0B37 1310 movb #32, 7(sp) ; make a leading sep. string
```

```
08 AE 04 0B AE 50 DD 0B57 1311      pushl r0      ; save size
04 AE 08 AE 50 09 0B59 1312      cvtsp r0,11(sp),#4,8(sp) ; cvrt exponent to packed
26 04 AE 04 36 0B60 1313      cvtpl #4,8(sp),4(sp) ; cvrt packed to long
03 15 0B66 1314      cmpl 4(sp),#38 ; see if exponent is huge
54 22 D0 0B6A 1315      bleq 60$ ; if not, br
5E 8E C0 0B6C 1316      movl #34,r4 ; plug max. huge prec.
8E 7C 0B6F 1317 60$:      addl (sp)+,sp ; clean off the stack
53 54 D0 0B72 1318      clrq (sp)+
37 37 BA 0B74 1319 100$:   movl r4,r3 ; return result in r3
05 05 0B77 1320      popr #^m<r0,r1,r2,r4,r5> ; restore regs
0B7A 1321      rsb
0B7A 1322
0B7A 1323
0B7A 1324      .end
```

PLI\$FORMAT
Symbol table

N 2

16-SEP-1984 02:18:05 VAX/VMS Macro V04-00
6-SEP-1984 11:37:47 [PLIRTL.SRC]PLIFORMAT.MAR;1

Page 25
(1)

ATR_M_VIRGIN	= 02000000		
ATR_V_FLTTRG	= 0000001A		
ATR_V_PRINT	= 00000007		
ATR_V_STRING	= 00000017		
BFORMATTAB	00000000	R	02
BITER	00000071	R	02
BLANK_FIELD	00000AB7	R	02
CHAR_FLT_CTX	00000AE3	R	02
COMEOF	0000017A	R	02
COMR	0000041A	R	02
COMRPAREN	0000048A	R	02
COMR_V2	000003F8	R	02
CVT_K_DST_ABIT	= 00000008		
CVT_K_DST_CHAR	= 00000005		
CVT_K_DST_FIXD	= 00000003		
CVT_K_DST_FLTD	= 00000004		
CVT_K_DST_PIC	= 00000000		
CVT_K_DST_VCHA	= 00000006		
CVT_K_SRC_ABIT	= 00000048		
CVT_K_SRC_CHAR	= 0000002D		
CVT_K_SRC_FIXB	= 00000009		
CVT_K_SRC_FIXD	= 0000001B		
CVT_K_SRC_FLTB	= 00000012		
CVT_K_SRC_FLTD	= 00000024		
CVT_K_SRC_PIC	= 00000000		
EXITER	000000C2	R	02
EXITER_COMMON	000000C6	R	02
EXITER_V2	000000B0	R	02
FAIL	00000526	R	02
FCB_B_ENVIR	000001C2		
FCB_B_ESA	0000012E		
FCB_B_EXTRA	0000003D		
FCB_B_FAB	000000A6		
FCB_B_IDENT	00000040		
FCB_B_IDENT_NAM	00000042		
FCB_B_NAM	000000F6		
FCB_B_NUMKCBS	0000003C		
FCB_B_RAB	00000062		
FCB_C_LEN	000001C2		
FCB_C_STRLIN	00000034		
FCB_L_ATTR	0000000C		
FCB_L_BUF	00000014		
FCB_L_BUF_END	00000018		
FCB_L_BUF_PT	0000001C		
FCB_L_CHDADDR	000001B2		
FCB_L_CONDIT	000001AE		
FCB_L_DTTR	00000010		
FCB_L_ERROR	00000008		
FCB_L_KCB	00000038		
FCB_L_NEXT	00000000		
FCB_L_PREVIOUS	00000004		
FCB_L_PRN	00000034		
FCB_Q_RFA	00000020		
FCB_W_COLUMN	0000002E		
FCB_W_IDENT_LEN	00000040		
FCB_W_LINE	00000030		
FCB_W_LINESIZE	0000002A		

FCB_W_PAGE	00000032		
FCB_W_PAGESIZE	0000002C		
FCB_W_REVISION	00000028		
FOR\$CVT_D_TE	*****	X	02
FOR\$CVT_G_TE	*****	X	02
FOR\$CVT_H_TE	*****	X	02
GETA	000001BA	R	02
GETB	000001DA	R	02
GETB1	000001CC	R	02
GETB2	000001D0	R	02
GETB3	000001D4	R	02
GETB4	000001D8	R	02
GETBITER	00000066	R	02
GETCOL	00000228	R	02
GETE	00000281	R	02
GETEOF	0000016E	R	02
GETEXPRITER	000000B6	R	02
GETEXPRITER_V2	000000A4	R	02
GETF	000002C0	R	02
GETITERCOM	000000D8	R	02
GETLITER	0000008F	R	02
GETP	000003B2	R	02
GETR	0000040E	R	02
GETRPAREN	0000047E	R	02
GETR_V2	000003EC	R	02
GETSKIP	00000451	R	02
GETWITER	0000007A	R	02
GETX	0000046D	R	02
GET_FORMAT_COM	000004A2	R	02
GET_FORMAT_PARM	000004A0	R	02
GET_FORMAT_PARM_1	0000049B	R	02
INVFRM	0000019C	R	02
INVFRMPRM	000001A6	R	02
INVSTRFMT	000001B0	R	02
LITER	0000009A	R	02
PIC\$B_BYTE_SIZE	= 00000004		
PIC\$W_PQ	= 00000000		
PLI\$CHRBITN_R6	*****	X	02
PLI\$GETFMT_R6	0000001E	RG	02
PLI\$GETNEDI_R6	*****	X	02
PLI\$GETSKIP_R2	*****	X	02
PLI\$GETSKIP1_R2	*****	X	02
PLI\$PUTFMT_R6	0000053C	RG	02
PLI\$PUTLINE_R6	*****	X	02
PLI\$PUTNEDI_R6	*****	X	02
PLI\$PUTPAGE_R6	*****	X	02
PLI\$PUTSKIP_R2	*****	X	02
PLI\$PUTSKIP1_R2	*****	X	02
PLI\$CHARFIXD_R6	*****	X	02
PLI\$CVRT CG_R3	*****	X	02
PLI\$FCHRFLTD_R6	*****	X	02
PLI\$FIXDCHAR_R6	*****	X	02
PLI\$FIXDFIXD_R6	*****	X	02
PLI\$IO_ERROR	*****	X	02
PLI\$VACID PIC	*****	X	02
PLI\$CNVERR	*****	X	02
PLI\$ERROR	*****	X	02

PLISFORMAT
Symbol table

B 3

16-SEP-1984 02:18:05 VAX/VMS Macro V04-00
6-SEP-1984 11:37:47 [PLIRTL.SRC]PLIFORMAT.MAR;1

Page 26
(1)

PLIS_FORMATOVFL	*****	X	02
PLIS_INVFMT Parm	*****	X	02
PLIS_INVFORMAT	*****	X	02
PLIS_INVSTRFMT	*****	X	02
PLIS_NOTPRINT	*****	X	02
PLIS_STROVFL	*****	X	02
PUTA	00000584	R	02
PUTB	000005EB	R	02
PUTB1	000005DD	R	02
PUTB2	000005E1	R	02
PUTB3	000005E5	R	02
PUTB4	000005E9	R	02
PUTBITER	0000006B	R	02
PUTCOL	00000748	R	02
PUTE	00000789	R	02
PUTEOF	00000174	R	02
PUTEXPRITER	000000BC	R	02
PUTEXPRITER_V2	000000AA	R	02
PUTF	000008DD	R	02
PUTLINE	000009FC	R	02
PUTLITER	00000094	R	02
PUTP	00000A1F	R	02
PUTPAGE	00000A46	R	02
PUTR	00000414	R	02
PUTRPAREN	00000484	R	02
PUTR_V2	000003F2	R	02
PUTSRIP	00000A66	R	02
PUTTAB	00000A7D	R	02
PUTWITER	0000007F	R	02
PUTX	00000AB4	R	02
RECOM	0000041E	R	02
SFSL_SAVE_FP	= 0000000C		
SIZ...	= 00000001		
STR_B_FIELD	00000018		
STR_C_LEN	00000C08		
STR_L_FLD_END	00000014		
STR_L_FLD_PT	00000010		
STR_L_FP	00000004		
STR_L_FS	0000000C		
STR_L_PARENT	00000008		
STR_L_SP	00000000		
STR_L_STACK	00000C04		
STR_L_STACK_END	00000408		
STR_M_BLANKEND	= 00000008		
STR_M_GFLOAT	= 00000010		
STR_M_MISSING	= 00000001		
STR_V_BLANKEND	= 00000003		
STR_V_GFLOAT	= 00000004		
STR_V_MISSING	= 00000000		
WITER	00000085	R	02
ZERO	000002CA	R	02

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000C08 (3080.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PLISCODE	00000B7A (2938.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.08	00:00:01.23
Command processing	73	00:00:00.51	00:00:03.45
Pass 1	210	00:00:08.45	00:00:26.43
Symbol table sort	0	00:00:00.74	00:00:01.18
Pass 2	244	00:00:03.09	00:00:06.34
Symbol table output	19	00:00:00.16	00:00:00.52
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	557	00:00:13.06	00:00:39.18

The working set limit was 1350 pages.
49255 bytes (97 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 338 non-local and 154 local symbols.
1324 source lines were read in Pass 1, producing 21 object records in Pass 2.
19 pages of virtual memory were used to define 17 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1	7
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	13

295 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS\$:PLIFORMAT/OBJ=OBJ\$:PLIFORMAT MSRC\$:PLIFORMAT/UPDATE=(ENH\$:PLIFORMAT)+LIB\$:PLIRTM

0308 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PLIFORMAT
LIS

PLIGETBUF
LIS

PLMSGTXT
LIS

PLIGETEDT
LIS

PLIHEEP
LIS

PLIPUTFIL
LIS

PLIRMSBIS
LIS

PLIRECOPT
LIS

PLIOPEN
LIS

PLIREAD
LIS

PLIPROTEC
LIS

PLIREWRT
LIS

PLIGETLIS
LIS

PLIPUTEDT
LIS

PLIPKDIUL
LIS

PLIPUTLIS
LIS

PLMSGPTR
LIS

PLIPKDIUS
LIS

PLIPUTBUF
LIS

PLIGETFIL
LIS